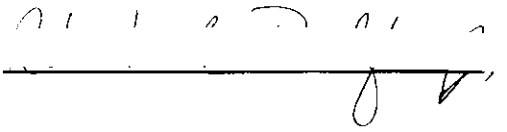
In presenting the dissertation as a partial fulfillment of the requirements for an advanced degree from the Georgia Institute of Technology, I agree that the Library of the Institute shall make it available for inspection and circulation in accordance with its regulations governing materials of this type. I agree that permission to copy from, or to publish from, this dissertation may be granted by the professor under whose direction it was written, or, in his absence, by the Dean of the Graduate Division when such copying or publication is solely for scholarly purposes and does not involve potential financial gain. It is understood that any copying from, or publication of, this dissertation which involves potential financial gain will not be allowed without written permission.



7/25/68

AN ALGORITHM FOR THE PLANT LOCATION PROBLEM

A THESIS

Presented to

The Faculty of the Division of Graduate

Studies and Research

by

Robert Lee Bulfin, Jr.

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in the

School of Industrial and Systems Engineering

Georgia Institute of Technology

March, 1972

AN ALGORITHM FOR THE PLANT

LOCATION PROBLEM

Approved:

Chairman

Date approved by Chairman: 2/29/72

ACKNOWLEDGMENTS

I wish to express my gratitude to Dr. V. E. Unger, not only for his many contributions to this research, but also for his advice and assistance throughout my graduate program. His guidance, assistance and encouragement have been invaluable, and are greatly appreciated.

I would also like to thank Dr. J. J. Jarvis and Dr. D. Fyffe for their assistance and suggestions while serving on the reading committee, and Dr. W. W. Swart, who persuaded me to attend graduate school.

I would like to acknowledge the National Science Foundation, which helped support this work through NSF Grant GK-5533. I would like to give special thanks to Dr. R. N. Lehrer and the School of Industrial and Systems Engineering for both financial and personal support.

I also wish to thank my sister, Mrs. Dianne Wilson, for proofreading and typing the rough draft, and especially Mrs. Peggy Weldon, for the excellent job she has done in typing the final draft.

Finally, I would like to thank my parents, for without them, none of this would have been possible.

TABLE OF CONTENTS

LIST OF TABLES

SUMMARY

This dissertation considers the plant location problem, which involves the selection of a subset of plants to be opened from a set of possible plant locations. Each plant has a fixed cost and an upper bound on its production capability. Fixed costs are incurred when a plant is opened (i.e. allowed to produce) and transportation costs are incurred in shipping the product from plants to points of demand. The objective is to select the subset of plants to open which minimizes total cost, while satisfying demand.

A solution technique for this problem is presented which gives optimal solutions. The problem is partitioned into an integer problem and a linear problem by using Benders' procedure. An implicit enumeration scheme is then applied to the integer problem. The linear problem is only solved for those subsets of plants which pass certain feasibility and optimality tests, and then it is used to generate constraints which bound the objective functions of the integer problem. Methodology is also developed and implemented for combining these constraints to form a surrogate constraint which is "better" than the individual constraints.

Computational results for the algorithm are presented.

CHAPTER I

INTRODUCTION

Problem Description

This research is concerned with a class of problems known as the plant location problem. Other names for the problem are location-allocation, facility location, warehouse location or site selection problems. All of these problems involve the location of one or more sources to be used to satisfy demand at various destinations. The objective is to determine which sources should be selected, such that demand is satisfied in an optimal manner.

This research will be restricted to the selection of locations from a pre-determined set of possible locations. The terminology used will be that of locating plants (sources) to satisfy demands at warehouses (destinations). The objective will be to minimize total system cost.

There are many costs to be considered in plant location problems, however they can be thought of as either fixed or variable costs. Variable costs occur for each unit of product. Examples of such costs are production, storage and shipping costs. These costs may be summed to provide a single variable cost for each plant-warehouse pair.* Fixed costs are costs that do not depend on the number of units, and occur only if a plant location is selected. Examples of fixed costs are construction, leasing and overhead costs, and may be combined to provide a

*The algorithm presented in Chapter III will handle any piecewise-linear convex cost function.

single fixed cost for each plant.

The constraints which must be satisfied are, first, that demand at warehouses must be satisfied, and secondly, the production required of a plant cannot exceed the upper capacity of that plant. The solution should specify one or more plants to be used to satisfy demand, and a distribution schedule utilizing the specified plants.

Objectives

The objectives of this thesis are:

(1) To develop an algorithm which finds optimal solutions to the plant location problem, as defined above, based on Benders' partitioning scheme.

(2) To investigate computational aspects of the algorithm and determine areas where further work is warranted.

Mathematical Formulation

The problem discussed in Chapter I may be stated mathematically as

$$\text{minimize} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{i=1}^{m} f_i y_i$$

subject to:

$$\text{(P)} \qquad \begin{aligned} & \sum_{j=1}^{n} x_{ij} \leq M_i y_i \qquad i = 1,2,\ldots,m \\ & \sum_{i=1}^{m} x_{ij} \geq D_j \qquad j = 1,2,\ldots,n \\ & x_{ij} \geq 0 \qquad i = 1,2,\ldots,m;\ j = 1,2,\ldots,n \\ & y_i \text{ binary} \qquad i = 1,2,\ldots,m \end{aligned}$$

where

m is the number of possible plant locations

n is the number of warehouses

c_{ij} is the cost of one unit of product made at plant i and shipped to warehouse j

f_i is the fixed cost at plant i

M_i is the capacity of plant i

D_j is the demand at warehouse j

x_{ij} is the flow from plant i to warehouse j

y_i is a binary variable whose value is one if plant i is open, and zero if closed.

The objective is to minimize total cost, consisting of both variable and fixed costs. The first set of constraints requires the number of units shipped from each plant to be less than the plant capacity if the plant is open, or zero if the plant is closed. The second group of constraints guarantees that the demand at each warehouse will be satisfied. Finally, the flow from plant i to warehouse j must be non-negative.

Clearly this is a mixed-integer programming problem.

Literature Survey

Work done on the fixed charge location problem may be categorized into approximate solution techniques and exact solution techniques. Approximate solution procedures are aimed at finding good solutions but cannot guarantee optimality, whereas exact algorithms give optimal solutions. This research will be concerned with developing an exact algorithm

to solve the problem.

Approximate Solution Techniques

There have been many approximate (or heuristic) solution techniques presented in the literature. This research will present three of the most widely cited papers. These three approaches are representative of the available heuristic programs for the fixed charge location problem.

Kuehn-Hamburger. Kuehn and Hamburger [12] have developed a heuristic program for solving the warehouse location problem. They present the objective as determining the correct balance of warehouse costs to savings created by: (1) reducing transportation costs; (2) improving delivery times. The solution method used can be thought of as two distinct parts: (1) a main program which locates warehouses, and (2) a bump and shift routine which modifies solutions obtained by the main program.

The main program locates warehouses one at a time until no decrease in cost occurs. Three heuristics used in the main program are: (1) Locations of promise occur at or near concentrations of demand. (2) Near optimum warehousing systems can be developed by locating warehouses one at a time, where the warehouse added contributes the least cost to the entire system. (3) Only a small subset of all possible locations need be considered in detail at each stage to determine which warehouse is to be added.

The bump and shift routine modifies solutions of the main program by eliminating those warehouses that have become uneconomical as a result of locating other warehouses, and then attempts to interchange potential warehouses while keeping demand satisfied.

Manne. Manne [14] explores the use of a "one-point move" algorithm to solve the plant location problem. Once locations are determined, the problem may be solved as a standard transportation problem. Letting a vector of ones and zeros (representing open and closed plants respectively) define a unit hypercube, and starting with some arbitrary lattice point, the algorithm searches for the adjacent feasible lattice point that results in the greatest improvement. For example, if the vector (0 0 1 0) represents the initial lattice point, a search would be made among the vectors (1 0 1 0), (0 1 1 0) and (0 0 1 1) to determine which, if any, combination leads to improvement. If none do, the algorithm terminates, otherwise a point is chosen and the search is repeated.

Results for symmetrical problems are acceptable, but asymmetrical problems were far from the optimal solution.

Feldman, Lehrer and Ray. Feldman, Lehrer, and Ray [6] have extended and modified the Kuehn-Hamburger model. One extension is to allow economies of scale to affect warehousing costs over the entire range of warehousing sizes, rather than having a single vixed cost for opening a warehouse. That is, the warehouse cost is concave rather than the sum of a fixed cost and a linear operating cost. Feldman et al. point out that this will change the assignment of customers to warehouses that have previously been opened, by the Keuhn-Hamburger procedure.

The solution technique used has both an add routine and a drop routine. Add routines start with all warehouses unavailable, and successively "open" warehouses, while drop routines start with all warehouses

available and successively close them. Add routines must first reach feasibility, and this may override cost considerations. Therefore Feldman et al. start with all warehouses open and drop one at the time until a local optimum is reached. Solutions determined by the drop routine are often better than those obtained by the add routine.

Exact Solution Techniques

All algorithms discussed in this section are exact (i.e. yield global optimum solutions) solution techniques. The important features of each will be discussed.

Efroymson and Ray. Efroymson and Ray [4] have developed an algorithm for the simple plant location problem. The formulation of the simple plant location problem assumes that there are no capacity constraints on plants, that is, any one plant can satisfy demand at all warehouses. The algorithm is a branch and bound scheme (see Lawler and Wood [13]) in which the binary restrictions are dropped and the problem solved as a linear program. Then branching is performed on non-binary variables, using the following three rules.

(1) Find a minimum bound for the cost reduction of opening a plant. This is done by determining the decrease in transportation cost obtained by opening the plant, and subtracting the fixed cost associated with that plant. If this quantity is positive, the plant will be fixed open.

(2) If it is cheaper to supply a given demand from an open plant, do so.

(3) Find a minimum bound for the cost reduction of opening a plant, and, if negative, leave the plant closed.

Spielberg. Another algorithm for the simple plant location problem has been developed by Spielberg [17]. His formulation of the problem includes side conditions, or configuration constraints. A typical configuration constraint might be that only a certain number of plants are allowed to be open at any time, or if some particular plant is open, then another particular plant must be closed. Benders' partitioning procedure [1] was applied and the problem partitioned into linear and integer sub-problems. The integer problem is then solved by an implicit enumeration scheme (see Geoffrion [8]), while the linear problem is used for bounding the integer problem. Branching rules used are bounding the allowable fixed cost, cancellation for non-positive gain (similar to Efroymson and Ray's rule one) and finally, the integrality of the solutions to the linear problem.

Sa. Sa [16] presents both a heuristic and an exact algorithm for the plant location problem (with capacity constraints). The exact algorithm uses a branch and bound technique, solving linear approximation problems to determine bounds on the mixed integer problem. When all free variables are set to one, the linear problem gives an upper bound, and by letting the integer variables assume fractional values, a lower bound is obtained.

The major pruning rules used, are one based on rule one of Efroymson and Ray, and one which bounds the fixed cost, as suggested by Gray [11].

Marks. Marks [15] has also applied branch and bound technique to the plant location problem. The main feature of his algorithm is the formulation of the problem as a network flow problem. The fixed cost of a plant is divided by the capacity of the plant, and the resulting cost

per unit is incurred for each unit a plant supplies. If a plant is not used at all, or it is used at capacity, then this approximation is equivalent to the fixed cost. If some plant supplies less than its capacity, but more than zero, the problem is then solved with the plant open and its fixed cost incurred, and again with the plant closed. This continues until the optimal solution is found and verified. This procedure utilizes the restart capabilities of network flow algorithms.

Davis and Ray. Another exact algorithm for the plant location problem has been developed by Davis and Ray [3]. This procedure is a branch and bound algorithm, and is similar to that of Marks. The original problem formulation is slightly different, and Davis and Ray use decomposition to obtain a master problem and a sub-problem. Like Marks, Davis and Ray solve sub-problems by a network flow algorithm. The master problem need only be solved a few times, but the subproblems are solved many times.

Ellwein. Ellwein [5] has developed an algorithm for the plant location problem with configuration constraints. Benders' partitioning is applied to the problem to reduce the mixed integer problem to two sub-problems, a linear problem and an integer problem. The integer problem is solved by an implicit enumeration scheme, using the linear problem to add bounds to the integer problem. Ellwein permanently opens plants (in a manner that is analogous to rule one of Efroymson and Ray) and uses the previously mentioned bounding of fixed cost of Gray.

CHAPTER II

MATHEMATICAL BACKGROUND

Partitioning the Problem

In order to solve (P), any mixed-integer programming algorithm could be used. However, this research will use the partitioning scheme suggested by Benders [1]. Thus (P) will be partitioned into two sub-problems, a linear problem (LP), and a zero-one integer problem (IP).

Following the method of Benders, the linear sub-problem for a fixed set of $\bar{y}_i$'s is given by

$$\text{minimize} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} f_i\bar{y}_i$$

subject to:

$$\text{(LP)} \qquad \begin{aligned} &\sum_{j=1}^{n} x_{ij} \le M_i\bar{y}_i \quad i = 1,2,\ldots,m \\ &\sum_{i=1}^{m} x_{ij} \ge D_j \quad j = 1,2,\ldots,n \\ &x_{ij} \ge 0 \quad i = 1,2,\ldots,m;\ j = 1,2,\ldots,n \end{aligned}$$

where all variables are defined in (P), and $\bar{y}_i$ are elements of a given binary vector.

Rearranging the constraints of (P) gives

$$-\sum_{j=1}^{n} x_{ij} + M_i y_i \geq 0 \quad i = 1,2,\ldots,m$$

$$\sum_{i=1}^{m} x_{ij} - D_j \geq 0 \quad j = 1,2,\ldots,n \ .$$

Summing all constraints gives

$$\sum_{i=1}^{m}\left(-\sum_{j=1}^{n} x_{ij} + M_i y_i\right) + \sum_{j=1}^{n}\left(\sum_{i=1}^{m} x_{ij} - D_j\right) \geq 0$$

which reduces to

$$\sum_{i=1}^{m} M_i y_i > \sum_{j=1}^{n} D_j \ .$$

$$y_i \text{ binary} \quad i = 1,2,\ldots,m \ .$$

This constraint requires sufficient capacity available to satisfy demand, without regard to a distribution schedule. Clearly, any feasible solution to (P) satisfies the composite constraint. Conversely, for any bindary solution to the composite constraint, a feasible solution to (P) can be constructed.

Now consider the following problem.

Minimize z_o

subject to:

(IP)
$$z_o \geq \sum_{j=1}^{n} \lambda_j^k D_j + \sum_{i=1}^{m} (f_i - M_i \pi_i^k) y_i \text{ all } \pi_i^k \lambda_j^k \in K$$

$$\sum_{i=1}^{m} M_i y_i \geq \sum_{j=1}^{n} D_j$$

---

†Constraints of this type will, henceforth, be called objective function constraints.

$$y_i \text{ binary} \qquad i = 1,2,\ldots,m$$

where K is the set of extreme points of U, and

$$U = \{\pi_i, \lambda_j \mid -\pi_i + \lambda_j \leq c_{ij}\} \ ,$$

and c_{ij} is the variable cost in (P).

Proposition 1

Let y_i^*, $i = 1,2,\ldots,m$ be the optimal solution to (IP) with solution value z_o^*. Then there exists x_{ij}^*, $i = 1,2,\ldots,m$; $j = 1,2,\ldots,n$ such that the optimal solution to (P) is y_i^*, x_{ij}^* $i = 1,2,\ldots,m$; $j = 1,2,\ldots,n$ with solution value

$$\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}x_{ij}^* + \sum_{i=1}^{m} f_i y_i^* = z_o^* \ .$$

Proof:

Substituting y_i^* in (P) and omitting the constant term in the objective function gives

$$\text{minimize} \qquad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}x_{ij}$$

subject to:

$$\sum_{j=1}^{n} x_{ij} \leq M_i y_i^* \qquad i = 1,2,\ldots,m$$

$$(P') \qquad \sum_{i=1}^{m} x_{ij} \geq D_j \qquad j = 1,2,\ldots,n$$

$$x_{ij} \geq 0 \qquad i = 1,2,\ldots,m; \ j = 1,2,\ldots,n.$$

The dual of (P') is

$$\text{maximize} \qquad -\sum_{i=1}^{m} M_i y_i^* \pi_i + \sum_{j=1}^{n} D_j \lambda_j$$

subject to:

$$-\pi_i + \lambda_j \leq c_{ij} \quad i=1,2,\ldots,m;\ j=1,2,\ldots,n$$

$$\pi_i \geq 0 \quad i=1,2,\ldots,m$$

$$\lambda_j \geq 0 \quad j=1,2,\ldots,n.$$

Clearly $\pi_i = \lambda_j = 0$ is a feasible solution to this problem, since $c_{ij} \geq 0$. Further, since

$$\sum_{i=1}^{m} M_i y_i^* \geq \sum_{j=1}^{n} D_j$$

it can be shown that (P') has a feasible solution. Therefore, since both problems have feasible solutions, both problems have finite optimal solutions. Letting x_{ij}^*, $i=1,2,\ldots,m$; $j=1,2,\ldots,n$ be the optimal solution to (P'), and adding the constant term

$$\sum_{i=1}^{m} f_i y_i^*$$

to both objective functions gives

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} x_{ij}^* + \sum_{i=1}^{m} f_i y_i^* =$$

$$\max_{\pi_i^k \lambda_j^k \in K} \; -\sum_{i=1}^{m} M_i y_i^* \pi_i^k + \sum_{j=1}^{n} D_j \lambda_j^k + \sum_{i=1}^{m} f_i y_i^*$$

where K is as defined above. The right hand side of the previous equation can be rewritten as

$$\max_{\pi_i^k \lambda^k \in K} \sum_{i=1}^{m} (f_i - M_i \pi_i^k)\, y_i^* + \sum_{j=1}^{n} \lambda_j^k D_j$$

which is the same as z_o^*, and hence

$$z_o^* = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}^* + \sum_{i=1}^{m} f_i y_i^* .$$

Then x_{ij}^*, y_i^*, $i = 1,2,\ldots,m$; $j=1,2,\ldots,n$ is a feasible solution to (P). Suppose it is not optimal. Then there must exist x_{ij}^{**}, y_i^{**}, $i=1,2,\ldots,m$; $j = 1,2,\ldots,n$ such that y_i^{**}, $i = 1,2,\ldots,m$ is feasible to (IP) and

$$\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}^{**} + \sum_{i=1}^{m} f_i y_i^{**} < \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}^{*} + \sum_{i=1}^{m} f_i y_i^{*} = z_o^*$$

and it must also be an optimal solution to (P'), since, if it were not, x_{ij}^{**} $i = 1,2,\ldots,m$; $j = 1,2,\ldots,n$ could not be optimal to (P). Thus

$$\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}^{**} + \sum_{i=1}^{m} f_i y_i^{**} =$$

$$\max_{\pi^k, \lambda^k \in K} \; - \sum_{i=1}^{m} M_i y_i^{**} \pi_i^k + \sum_{j=1}^{n} D_j \lambda_j^k + \sum_{i=1}^{m} f_i y_i^{*},$$

and since y_i^{**}, $i = 1,2,\ldots,m$ is feasible to (P) it is also feasible to (IP) and thus a solution to (IP) can be found with value $z_o^{**} < z_o^*$. But this contradicts the assumption that z_o^* was the optimal solution to (IP),

and the proposition is demonstrated.

In order to solve (P), Benders suggests alternately solving (LP) and (IP) until the solution values are equal. Thus, Proposition 1 guarantees that this procedure will produce the optimal solution to (P), and since K is finite, there will be a finite number of steps. This does not fully exploit the properties of (P) however, and a more efficient procedure, based on the work of Unger [19], will be explored.

Surrogate Constraints

Since the number of objective function constraints of (IP) may become large, it would be beneficial to condense as much information as possible into one constraint. In particular, if the constraints are only enforced individually, then it is likely that all of them will not be satisfied when considered as a group. A constraint which combines information from many constraints is called a composite or surrogate constraint. Glover [10] and Geoffrion [9] have been instrumental in the development of the concepts of "strongest" surrogate constraints.

Suppose some of the y_i's are restricted to be one or zero. Let S be the set associated with restricted, or fixed variables so that if $y_i = 1$ then $+i \in S$, and if $y_i = 0$ then $-i \in S$. Furthermore, let z^+, called the incumbent solution, be the value of the best solution found thus far, and let ℓ be the number of objective function constraints added thus far.

Now the integer problem (IP) may be written as

minimize z_o

subject to:

$$z_o - b^k - \sum_{i \notin S} a_i^k y_i \geq 0 \quad k = 1, 2, \ldots, \ell$$

$$
(IP') \qquad
\begin{aligned}
&\sum_{i \notin S} M_i y_i - \bar{M} \geq 0 \\
&y_i \text{ binary} \quad i \notin S \\
&z_o \text{ unrestricted}
\end{aligned}
$$

where

$$\bar{b}^k = \sum_{j=1}^{k} {}^{k}D_j + \sum_{\lambda i \in S} a_i y_i \qquad k=1,2,\ldots,\ell$$

$$a_i^k = f_i - \pi_i^k M_i \qquad i=1,2,\ldots,m;\ k=1,2,\ldots,\ell$$

and

$$\bar{M} = \sum_{j=1}^{n} D_j - \sum_{+i \in S} M_i$$

Taking a convex combination of the constraints of (IP'), any solution to (IP') which is better than the incumbent solution must also satisfy

$$\sum_{k=1}^{\ell} (z_o - \bar{b}^k - \sum_{i \notin S} a_i^k y_i)\mu_k + (\sum_{i \notin S} M_i y_i - \bar{M})\mu_{\ell+1}$$

$$+ (z^+ - z_o) > 0$$

$$\sum_{k=1}^{\ell} \mu_k = 1$$

$$\mu_k \geq 0 \qquad k = 1,2,\ldots,\ell+1$$

$$y_i \text{ binary} \quad i \notin S$$

$$z_o \text{ unrestricted.}$$

The definition of the "strength" of a surrogate constraint is now made, which is analogous to that made by Geoffrion. That is, the surrogate constraint

$$\sum_{k=1}^{\ell} (z_o - \bar{b}^k - \sum_{i \notin S} a_i^k y_i)\mu_k^1 + (\sum_{i \notin S} M_i y_i - \bar{M})\mu_{\ell+1}^1 + (z^+ - z_o) > 0$$

$$\sum_{k=1}^{\ell} \mu_k = 1$$

is said to be stronger than the surrogate constraint

$$\sum_{k=1}^{\ell} (z_o - \bar{b}^k - \sum_{i \notin S} a_i^k y_i)\mu_k^2 + (\sum_{i \notin S} M_i y_i - \bar{M})\mu_{\ell+1}^2 + (z^+ - z_o) > 0$$

$$\sum_{k=1}^{\ell} \mu_k = 1$$

if the maximum of the left-hand side of the first is less than the maximum of the left-hand side of the second constraint.

Thus, finding the strongest surrogate constraint is, then, finding μ_k, $k = 1,2,\ldots,\ell$ and y_i, $i = 1,2,\ldots,m$ satisfying

$$\min_{\mu} \max_{y} \sum_{k=1}^{\ell} (z_o - \bar{b}^k - \sum_{i \notin S} a_i^k y_i)\mu_k + (\sum_{i \notin S} M_i y_i - \bar{M})\mu_{\ell+1} + (z^+ - z_o)$$

subject to:

$$\sum_{k=1}^{\ell} \mu_k = 1$$

$$\mu_k \geq 0 \quad k = 1,2,\ldots,\ell+1$$

$$y_i \text{ binary} \quad i \notin S$$

z_o unrestricted

Since $\sum_{k=1}^{\ell} \mu_k = 1$ then $\sum_{k=1}^{\ell} z_{o\,k} = z_o$, and z_o can be removed from the objective function.

Now rearranging terms gives

$$z^+ + \min_{\mu_k} \sum_{k=1}^{\ell} (-\bar{b}^k \mu_k) - \bar{M} \mu_{\ell+1}$$

$$+ \max_{y_i} \sum_{i \notin S} (M_i \mu_{\ell+1} - \sum_{k=1}^{\ell} a_i^k \mu_k) y_i$$

subject to:

$$\sum_{k=1}^{\ell} \mu_k = 1$$

$$\mu_k \geq 0 \quad k = 1,2,\ldots,\ell+1$$

$$y_i \text{ binary} \quad i \notin S$$

Now for given μ_k, $k = 1,2,\ldots,\ell$

$$\max_{y_i} \sum_{i \notin S} (M_i \mu_{\ell+1} - \sum_{k=1}^{\ell} a_i^k \mu_k)\, y_i$$

$$y_i \text{ binary} \quad i \notin S$$

$$= \max_{y_i} \sum_{i \notin S} \left(M_i \mu_{\ell+1} - \sum_{k=1}^{\ell} a_i^k \mu_k\right) y_i$$

$$0 \leq y_i \leq 1 \quad i \notin S$$

From duality, the last program may be expressed as

minimize $$\sum_{i \notin S} \omega_i$$

subject to:

$$-\omega_i - \sum_{k=1}^{\ell} a_i^k \mu_k + M_i \mu_{\ell+1} \leq 0 \quad i \notin S$$

$$\omega_i \geq 0 \quad i \notin S$$

Substituting into the original problem gives

minimize $$z^+ - \sum_{k=1}^{\ell} \bar{b}^k \mu_k - \bar{M} \mu_{\ell+1} + \sum_{i \notin S} \omega_i$$

subject to:

$$-\omega_i - \sum_{k=1}^{\ell} a_i^k \mu_k + M_i \mu_{\ell+1} \leq 0 \quad i \notin S$$

$$\sum_{k=1}^{\ell} \mu_k = 1$$

$$\mu_k \geq 0 \quad k = 1, 2, \ldots, \ell+1$$

$$\omega_i \geq 0 \quad i \notin S$$

The solution of this problem gives weights (μ_k) for each existing constraint to form a surrogate constraint.

Taking the dual of (SP) gives

minimize δ

subject to:

$$\delta - \sum_{i \notin S} a_i^k \gamma_i - \bar{b}^k \geq 0 \quad k = 1, 2, \ldots, -$$

$$\sum_{i \notin S} M_i \gamma_i - \bar{M} \geq 0$$

$$0 \leq \gamma_i \leq 1 \quad i \notin S$$

δ unrestricted.

It is obvious that (DSP) is the same problem as (IP') with the binary restriction relaxed.

If the capacity constraint

$$\sum_{i \in S} M_i \gamma_i \geq \sum_{j=1}^{n} D_j$$

is satisfied for the set of fixed variables S, then $\bar{M} \leq 0$ and $\mu_{\ell+1}$ will equal zero in the optimal solution to (SP). Thus the surrogate problem is

$$\text{minimize} \quad z^+ - \sum_{k=1}^{\ell} \bar{b}^k \mu_k + \sum_{i \notin S} \omega_i$$

subject to:

$$-\omega_i - \sum_{k=1}^{\ell} a_i^k \mu_k \leq 0 \quad i \notin S$$

$$\sum_{k=1}^{\ell} \mu_k = 1$$

$$\mu_k \geq 0 \quad k = 1,2,\ldots,\ell$$

$$\omega_i \geq 0 \quad i \notin S \ .$$

CHAPTER III

STATEMENT OF THE ALGORITHM

An algorithm for the solution of the plant location problem (P) can now be stated. To facilitate discussion of the algorithm, the following definitions are made.

Let

$$b^k = \sum_{j=1}^{n} \lambda_j^k D_j \quad k = 1,2,\ldots,\ell$$

where λ_j^k is the dual variable associated with the $j^{\underline{th}}$ warehouse in the $k^{\underline{th}}$ solution of (LP), and D_j is the demand at warehouse j.

Also let

$$a_i^k = f_i - M_i\pi_i^k \quad i = 1,2,\ldots,m; \quad k = 1,2,\ldots,\ell$$

where π_i^k is the dual variable associated with the $i^{\underline{th}}$ plant in the $k^{\underline{th}}$ solution of (LP), and f_i, M_i are the fixed cost and capacity of plant i.

(IP) may now be rewritten as

minimize z_o

subject to:

$$z_o \geq b^k + \sum_{i=1}^{m} a_i^k y_i \quad k = 1,2,\ldots,\ell$$

$$\sum_{i=1}^{m} M_i y_i \geq \sum_{j=1}^{n} D_j$$

y_i binary $\quad i = 1,2,\ldots,m$.

Suppose that some subset of the possible plant locations must be either opened or closed. This is represented by a set S, called a partial solution, which contains the positive value of the subscript if the plant must be open, and the negative value if it is restricted to be closed. If ±i is in S, then the plant is said to be fixed, if not then the plant is unassigned or free.

Let z^+ be the current best feasible solution, and TC^+, FC^+ and y_i^+, $i = 1,2,\ldots,m$ represent the corresponding values of the transportation cost, fixed cost and the binary portion of the solution.

The algorithm is:

Step 0: Considering only the transportation cost (i.e. disregarding the fixed costs), set up and solve (LP) (see Appendix A for details) with all plants "open." If a plant is used, include its index in S. This solution of (LP), say TMIN, is the lowest possible transportation cost for any partial solution S, since all plants are available for use. Now define the sum of the fixed costs for available plants as

$$FS = \sum_{+i \in S} f_i$$

and the total available capacity for this solution is

$$MS = \sum_{+i \in S} M_i \ .$$

The binary portion of the solution vector and the new incumbent solution value are

$$y_i^+ = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{otherwise} \end{cases} \quad i = 1,2,\ldots,m \ .$$

and

$$z^+ = TMIN + FS \ .$$

Furthermore, set

$$y_i' = y_i^+ \qquad i = 1,2,\ldots,m \ .$$

Now set $\ell = 1$, and add an objective function constraint, formed from the dual variables of this solution to (LP). If $a_i^1 \leq 0$, then plant i must be "open" in an optimal solution to (P).† (This is true only for the constraint formed from the dual variables of (LP) when all plants are available for use.) This is true, since $a_i^1 \leq 0$, implies $\pi_i^1 M_i \geq f_i$ and thus the lower bound on the increase in transportation cost for not having that plant available is at least as great as the fixed cost of "opening" the plant. Adjust S such that permanently "opened" plants are in the left-most positions, and underlined.

Perform the following for each plant i that is not permanently fixed open:

Solve (LP) with all plants available for use except plant i, and denote its solution value by $TCOST_i$. (Again the solution represents only the transportation cost of (LP), disregarding the fixed costs of opening plants.) Clearly $TCOST_i \geq TMIN$. Now let

$$ITCOST_i = TCOST_i - TMIN \ .$$

This is the increase in transportation cost which occurs when plant i is

---

† This result is due to Ellwein.

unavailable for use. If

$$ITCOST_i \geq f_i$$

then the incremental transportation cost exceeds the cost of "opening" the plant, and the plant may be permanently "opened" as before.†

Furthermore, if the solution is better than the incumbent, the current solution can replace the incumbent solution, and an objective function constraint formed from the dual variables of each solution to (LP) is added to the list of objective function constraints.

Step 1: If

$$FS + TMIN + \sum_{-i\in S} ITCOST_i \geq z^+$$

then the cost incurred by this set of fixed variables is greater than the incumbent solution. Therefore go to step seven and backtrack.

For each objective function constraint generated thus far, find the completion of the partial solution S which minimizes its value. Denote the value of this completion of the $k^{\underline{th}}$ objective function constraint by z^k. To find this completion let

$$y_i' = \begin{cases} 1 & \text{if } +i\in S \\ 0 & \text{otherwise} \end{cases} \quad i = 1,2,\ldots,m,$$

$$FS = \sum_{+i\in S} f_i ,$$

$$MS = \sum_{+i\in S} M_i .$$

---

†This result is due to Ellwein.

Define ℓ sets $Q^1, Q^2, \ldots, Q^\ell$ as follows:

$$Q^k = \{ i \mid i \notin S,\ a_i^k < 0,\ TMIN + FS + f_i < z^+ \} \quad k = 1, 2, \ldots, \ell.$$

Clearly the best completion value of each objective function constraint is

$$z^k = b^k + \sum_{+i \in Q^k \cup S} a_i^k \quad k = 1, 2, \ldots, \ell$$

If any $z^k \geq z^+$, go to step seven.

Step 2: Find the maximum z^k and denote its value by z' and its index by k'. Let

$$\bar{FS} = \sum_{i \in Q^{k'}} f_i,$$

$$\bar{MS} = \sum_{i \in Q^{k'}} M_i$$

and

$$y_i' = 1 \qquad i \in Q^{k'}.$$

If $MS + \bar{MS} \geq \sum_{j=1}^{n} D_j$, the completion can satisfy demand. Go to step four.

Step 3: Let R denote the set of candidates to include in S, such that total capacity is increased, while retaining a possible solution better than the incumbent. Let

$$R = \{ i \mid i \notin S,\ y_i' = 0,\ TMIN + FS + f_i < z^+ \}$$

If $R = \emptyset$, then there exists no plant, which, when "opened," can possibly produce a feasible solution better than the incumbent, therefore go to step seven and backtrack.

If $R \neq \emptyset$, then choose some i_o in R such that the ratio f_i/M_i is minimized. Then place $+i_o$ in S and if there is still not enough capacity available to satisfy demand, set $R = \emptyset$ and repeat step three, otherwise return to step one.

Step 4: Find the value of each objective function constraint using the completion $Q^{k'}$ as found in step two. Denote it as

$$\bar{z}^k = b^k + \sum_{i=1}^{m} a_i^k y_i' \quad k = 1, 2, \ldots, \ell .$$

Denote the maximum $\bar{z}^k$ by $\bar{z}'$ and denote its index by $\bar{k}$.

If $z' = \bar{z}'$, then the most binding constraint $\bar{k}$ has the same value as the worst of the best completions, therefore go to step six and solve (LP).

Step 5: Try to resolve the discrepancy between z' and $\bar{z}'$. This can be done by examining free variables, which, when complemented, decrease the value of $\bar{z}'$, quite possibly at the expense of z'. Define the sets T^+ and T^- as follows:

$$T^+ = \{ i \mid i \notin S, \quad y_i' = 1, \quad a_i^{\bar{k}} > 0 \}$$
$$T^- = \{ i \mid i \notin S, \quad y_i' = 0, \quad a_i^{\bar{k}} < 0 \} .$$

There are two conditions that may occur. If $T^+ \bigcup T^- = \emptyset$ then go to (a). If $T^+ \bigcup T^- \neq \emptyset$ then go to (b).

(a) $T^+ \bigcup T^- = \emptyset$.

If $\bar{z}' \geq z^+$, there exists no better solution with this partial solution S, therefore go to step seven and backtrack.

Otherwise solve (SP), to determine the surrogate constraint. If the solution value of (SP) is less than zero, Geoffrion [9] has shown that there does not exist a completion to S better than the incumbent solution, therefore go to step seven and backtrack. If the optimal solution to (SP) has value greater than zero, and the dual variables corresponding to the free variables in (IP) are binary, the optimal solution to (P) for the given S has been found. Record z^+, TC^+, FC^+ and y_i^+, and go to step seven and backtrack. If some of the dual variables are not binary, increment ℓ and add the "strongest" surrogate constraint,

$$z_o \geq \sum_{k=1}^{\ell} b^k \mu_k^* + \sum_{i=1}^{m} \sum_{k=1}^{\ell} a_i^k \mu_k^* \; y_i$$

where μ_k^*, $k = 1,2,\ldots,\ell$ is the solution to (SP).

After the surrogate constraint is added, return to step one.

(b) If $I^+ \bigcup I^- \neq \emptyset$, then we can improve the value of the constraint $\bar{k}$ by the following:

There exists some plant, which when included in S, will decrease the value of $\bar{z}'$. Choose that i, say i_o, which gives the minimum of the following two expressions.

(1) $$\min_{i \in I^+} \max_{k} (\bar{z}^k - a_i^k)$$

or

(2) $$\min_{i \in I^-} \max_{k} (\bar{z}^k + a_i^k).$$

If $i_o \epsilon T^+$ then $-i_o$ is placed in S, and return to step one.

If $i_o \epsilon T^-$ then $+i_o$ is placed in S, set $y'_{i_o} = 1$ and return to step one.

Step 6: Solve (LP) with the current completion of S. Let the capacity of plant i be M_i if $y'_i = 1$ and zero otherwise. Denote the solution value to (LP) by z^*, with TC representing the transportation cost and FC the fixed cost. There are two conditions that may occur:

(a) If $z^* = \bar{z}'$, a solution exists to (P) that is better than the incumbent. Record the new values of z^+, TC^+, FC^+ and y_i^+. Thus S has been fathomed, since we have been working with the best completion of S, therefore go to step seven and backtrack.

(b) If $z^* \neq \bar{z}'$, an element of K not currently used in (IP) has been found, and an objective function constraint is added. Increment ℓ by one and let

$$b^\ell = \sum_{j=1}^{n} D_j \lambda_j^\ell$$

and

$$a_i^\ell = f_i - \pi_i^\ell M_i \quad i = 1,2,\ldots,m .$$

The constraint to be added is

$$z_o \geq b^\ell + \sum_{i=1}^{m} a_i^\ell y_i .$$

Furthermore, if $z^* < z^+$, a better solution to (P) has been found, and the new z^+, TC^+, FC^+ and y_i^+, $i = 1,2,\ldots,m$ are recorded, and return to step one.

Step 7: Find the rightmost non-underlined element of S. Complement and underline the element, and delete all elements to its right. Update FS, MS and y_i' accordingly and go to step one.

If no such element exists, the algorithm terminates with the incumbent solution being the optimal solution to (P).

Geoffrion [8] has shown that the backtracking procedure used in step seven leads to a non-redundant sequence of partial solutions which terminate only when all such solutions have been explicitly or implicitly explored. Since there are only a finite number of partial solutions, and a finite number of elements in the set K, the algorithm will terminate in a finite number of iterations. Further, since a backtrack is performed only when a partial solution cannot become feasible or produce a solution better than the incumbent, the optimal solution will be reached.

CHAPTER IV

CONCLUSIONS AND RESULTS

Computational Experience

The algorithm presented in Chapter III has been coded in Fortran and run on a Univac 1108 computer. The Fortran code used is given in Appendix B and the problem data are presented in Appendix C.

Two algorithms were actually tried, one which generated surrogate constraints, and one which did not. Computational experience indicates that the algorithm which did not use surrogate constraints is superior to the algorithm which uses surrogate constraints. The results of several test problems are presented in Table 1.

The problems solved were also solved by Ellwein [5], although some were changed slightly. The changes were necessary since Ellwein included configuration constraints, and the algorithm presented in Chapter III was not intended to handle configuration constraints. The solution times for capacitated problems without configuration constraints were better than those obtained by Ellwein. However for uncapacitated problems, or problems with configuration constraints Ellwein's algorithm produced better solution times. His times were significantly better for those problems with configuration constraints, since the configuration constraints remove large portions of the solution space from consideration. Care must be taken when comparing the algorithms, since they were run on different computers (Ellwein used an IBM 360/67) and were coded by different programmers.

Table 1. Computational Results

Problem Number	Problem Size (mxn)	Number of Plants Open (Fixed/Optimal)	Cost ($1000) (Variable/Total)	Number of Times (LP) Solved	Solution Time (Minutes)
1	10x20	2/6	158/233	53	.28
2	15x45	5/11	348/641	18	.22
3	15x45	2/7	258/519	64	.94
4	15x45	1/6	187/382	182	3.18
5	15x45	1/4	190/317	127	1.60
6	15x45	0/4	454/1759	56	.78
7	16x50	11/12	961/1111	20	.48
8	16x50	11/12	961/1153	22	.53
9	16x50	7/9	878/978	115	1.51
10	16x50	7/9	878/978	118	1.66
11	16x50	3/-	-/1020	-	5.00 #
12	16x50	4/-	-/1063	-	5.00 #
13	25x50	10/-	-/805	-	5.00 #
14	25x50	0/-	-/1507	-	5.00 #
15	25x50	0/-	-/1486	-	5.00 #
16	25x50	0/3	1080/1690	132	4.95

#Run stopped at 5.00 minutes.

Extensions and Further Research

Algorithmic Extensions

Two immediate algorithmic changes which should reduce solution time will now be noted. First, no computational advantage was taken of the structure of the sub-problems (LP) and (SP). The out-of-kilter algorithm which was used to solve (LP) could be streamlined, since all lower bounds on arcs are zero, thereby eliminating some kilter states from consideration. Similarly, using as the initial basis the optimal basis of the previous solutions would have reduced the solution time for (SP).

There is also need for determining rules for discarding objective function constraints. The algorithm presented in Chapter III kept an arbitrary number (one hundred), and when that number was exceeded, replaced the older constraints with the new ones. There appears to be significant computational gains by reducing the number of constraints kept, while reducing the information given as little as possible.

Recall that the dual of the surrogate problem (DSP) was the same problem as (IP') with the binary restriction relaxed. Furthermore, solving (DSP) gives all of the information needed for the surrogate constraint tests. If (DSP) is solved, then, in addition to the surrogate constraint test information, penalties for rounding non-integer variable as suggested by Tomlin [18] can be obtained. If the solution value of (DSP) plus the penalty is greater than or equal to the incumbent solution, then a backtrack may be performed.

If this were implemented, replacing step one of the current algorithm by solving (DSP) and either backtracking, adding a surrogate

constraint or finding a better solution appears fruitful. This would require other changes in the algorithm, but would make more efficient use of the information obtained from the surrogate constraint problem.

A similar approach would be to collapse all of the objective function constraints of (IP) into a single constraint by the methods of Bradley [2]. This constraint could then be used as the objective function of a knapsack problem, with the capacity constraint as the only constraint. This could then be used to find completions to the partial solutions and incorporated in the algorithm of Chapter III.

Problem Extension

Due to the setting up of (LP) as a network flow problem, the extension of the algorithm presented in Chapter III to the fixed charge transportation problem appears to be straightforward. Rather than incurring a fixed cost only on arcs from the super source to the plants (see Appendix A), there would be a fixed cost on every arc. This would greatly increase the number of integer variables since there would now be one for every arc.

Another formulation of the plant location problem would be to allow plant capacities, transportation costs, demands and fixed charges to vary from time period to time period. This requires many more integer variables (for a t period problem with m plants it would require txm integer variables). Also, plants may be open one period and closed the next unless otherwise restricted. This extension would be much more difficult than would the extension to the fixed charge transportation problem.

Conclusions

The basic approach of the algorithm appears to be good, since the

solution times for capacitated plant location problems without side conditions are better than the published results of any other algorithm. Further work on uncapacitated problems, and the addition of side constraints should be done, along with the investigation of the areas noted previously.

APPENDIX A

SOLUTION OF SUB-PROBLEMS

Solution of (LP)

The problem

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} f_i\bar{y}_i$$

s.t.

$$\text{(LP)} \qquad \begin{aligned} &\sum_{j=1}^{n} x_{ij} \le M_i\bar{y}_i \qquad i = 1,2,\ldots,m \\ &\sum_{i=1}^{m} x_{ij} \ge D_j \qquad j = 1,2,\ldots,n \\ &x_{ij} \ge 0 \qquad i = 1,2,\ldots,m;\ \ j = 1,2,\ldots,n \end{aligned}$$

is simply a transportation problem, and could be solved by means of a transportation algorithm. It can also be considered a network flow problem. Since (LP) is being solved many times in the integer programming algorithm, the restart capabilities of the out-of-kilter algorithm are very beneficial. Therefore the problem is solved as a maximum flow, minimum cost network flow problem (see Ford and Fulkerson [7]).

The network representation of (LP) is given in Table 2. There are nodes corresponding to each plant (P_1, P_2,...,P_m), each warehouse (W_1, W_2,...,W_n), and two dummy nodes, a super plant (SP) and a super warehouse (SW). All lower capacities are zero.

The m arcs from SP to P_1, P_2,...,P_m are used only to insure that the flow out of each plant is less than the current capacity (i.e. M_i if

Table 2. The Network Representation of (LP)

Arc Number	Beginning Node	Ending Node	Upper Capacity	Cost per Unit
1	SP	P_1	M_1y_1	0
2	SP	P_2	M_2y_2	0
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
m	SP	P_m	M_my_m	0
m+1	P_1	W_1	∞	c_{11}
m+2	P_1	W_2	∞	c_{12}
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
m+n+1	P_2	W_1	∞	c_{21}
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
m + (i-1)*n+j	P_i	W_j	∞	c_{ij}
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
m +m*n +1	W_1	SW	D_1	0
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
m +m*n +n	W_n	SW	D_n	0
m +m*n +n +1	SW	SP	∞	$-\infty$

plant i is open, zero otherwise). Arcs from plants to warehouses are uncapacitated, and any flow through these arcs incurs the unit cost specified in (LP). Arcs from warehouses W_1, W_2,...,W_n to SW are used to bound the amount of flow sent to a warehouse. The arc from SW to SP is the return arc. Its large negative cost forces flow through the network until saturation occurs, and the out-of-kilter algorithm does this while keeping costs minimized.

The dual variables of the out-of-kilter algorithm can be obtained from the node numbers associated with each node. The absolute magnitude of the node numbers is unimportant, but the relative difference reflects the value of the dual variables of (LP). By starting all node numbers at zero, and restricting the node number of SP to be zero, the node numbers of P_1, P_2,..., P_m and W_1, W_2,...,W_n are exactly the dual variables of (LP).

Solution of (SP)

The problem of finding strongest surrogate constraints is

$$\text{maximize} \quad z^+ - \sum_{k=1}^{\ell} \mu_k b_k' + \bar{M}\mu_{\ell+1} - \sum_{i \notin S} W_i$$

$$\text{s.t.} \quad -W_i - \sum_{k=1}^{\ell} \mu_k a_i^k + M_i \mu_{\ell+1} \leq 0 \quad i \notin S$$

$$\sum_{k=1}^{\ell} \mu_k = 0$$

$$\mu_k \geq 0 \quad k = 1, 2, \ldots, \ell+1$$

$$W_i \geq 0 \quad i \notin S .$$

This is a linear programming problem, and can be solved by any of the many known methods. The solution technique used in this research is a standard simplex method. No advantage was taken of the computationally efficient techniques available, such as revised simplex.

A better solution technique might be to solve the dual of (SP). This would allow the use of a bounded variable algorithm, and also allow use of penalty functions as discussed in Chapter IV.

APPENDIX B

FORTRAN PROGRAM FOR PLANT LOCATION ALGORITHM

```
      COMMON  ITER,ISMJ,ICTR,BACK,SUMMI,SUMDJ,YBEST(30)
      COMMON NOSRCS,NODEST,NODES,ARCS,ISUM,LARGE,ZPLUS,TCPLUS,FCPLUS,
     *       YPLUS(30),COST(1376),F(1376),HI(1376),PI(79),FIX(30),
     *       AVAIL(30),DEM(50)
      COMMON NTFEAS
      COMMON L,ITAL,SS(30),CONST(100),CV(100,30),U(100),ITX
      LOGICAL NTFEAS,NOLP
      INTEGER NOSRCS,NODEST,NODES,ARCS,ISUM,LARGE,ZPLUS,TCPLUS,
     *       YPLUS,F,HI,PI,FIX,AVAIL,DEM,YBEST,COST,SUMMI,SUMDJ
      INTEGER UNDER(30),SV(30),SS,YS(30),
     *       ZFIX(100),ZBAR(100),CV,
     *       FXCOST,FXS,ZMAX,SUMFRE,FRE,SUMTS,ZBMAX,ZSTAR
      LOGICAL FIRST,SETONE,BACK
      REAL A,AMAX
      DIMENSION LOCMIN(30)
      INTEGER CONST,FCPLUS
    1 FORMAT ( )
      READ(5,1) NOSRCS,NODEST
      ARCS=NOSRCS+NOSRCS*NODEST+NODEST+1
      NODES=NOSRCS+NODEST+2
      LARGE=999999999
      II=NOSRCS+1
      IN=NOSRCS+NOSRCS*NODEST
      READ(5,1) (COST(I),I=II,IN)
      READ(5,1) ( FIX(I),I=1,NOSRCS)
      READ(5,1) (AVAIL(I),I=1,NOSRCS)
      READ(5,1) (DEM(J),J=1,NODEST)
      NTFEAS=.FALSE.
      SUMDJ=0
      DO 11 J=1,NODEST
      SUMDJ=SUMDJ+DEM(J)
   11 CONTINUE
      NOLP=.FALSE.
      SUMMI=0
      DO 2 I=1,NOSRCS
      SUMMI=SUMMI+AVAIL(I)
    2 CONTINUE
      IF(SUMMI.LT.SUMDJ) GO TO 8999
      ITAL=0
      SUMMI=0
      FXCOST=0
      FIRST=.TRUE.
      L=0
      ICTR=0
      ITX=0
      ITER=0
      NOTIME=0
      DO 5 I=1,NOSRCS
      UNDER(I)=0
      SV(I)=0
      SS(I)=0
    5 CONTINUE
C     SOLVE OUT-OF-KILTER WITH ALL PLANTS OPEN
      CALL NETWRK
```

```
      IF(NTFEAS) GO TO 8999
      DO 15 I=1,NOSRCS
      IF(F(I).GT.0) GO TO 12
      YS(I)=0
      YBEST(I)=0
      YPLUS(I)=0
      GO TO 15
   12 YS(I)=1
      YBEST(I)=1
      YPLUS(I)=1
      ITAL=ITAL+1
      FXCOST=FXCOST+FIX(I)
      SV(ITAL)=I
      SS(I)=1
      SUMMI=SUMMI+AVAIL(I)
   15 CONTINUE
      FXS=FXCOST
      FCPLUS=FXS
      MINTRN=ISUM
      TCPLUS=ISUM
      ZPLUS=FXS+ISUM
      IC=0
      DO 20 J=1,ITAL
      I=SV(J)
      II=PI(I)*AVAIL(I)
      IF(II.LT.FIX(I)) GO TO 20
      IC=IC+1
      SV(J)=SV(IC)
      SV(IC)=I
      UNDER(I)=1
   20 CONTINUE
   21 FORMAT(2X,'THE NUMBER OF FIXED VARIABLES IS ',I4)
      J=0
      GO TO 6500
   25 J=J+1
      IF(J.GT.NOSRCS) GO TO 100
      IF(UNDER(J).NE.0) GO TO 25
      HI(J)=0
      CALL NETWRK
      LOCMIN(J)=ISUM-MINTRN
      HI(J)=AVAIL(J)
      IFXX=0
      DO 40 I=1,NOSRCS
   40 IF(F(I).GT.0) IFXX=IFXX+FIX(I)
      II=IFXX+ISUM
      IF(II.GE.ZPLUS) GO TO 6500
      ZPLUS=II
      TCPLUS=ISUM
      FCPLUS=IFXX
      DO 70 I=1,NOSRCS
      IF(F(I).GT.0) GO TO 50
      YPLUS(I)=0
      GO TO 70
   50 YPLUS(I)=1
```

```
 70 CONTINUE
    GO TO 6500
100 CONTINUE
    FIRST=.FALSE.
    DO 200 I=1,NOSRCS
    IF(LOCMIN(I).LT.FIX(I).OR.UNDER(I).NE.0) GO TO 200
    II=IC+1
    DO 150 J=II,ITAL
    IF(ABS(SV(J)).NE.I) GO TO 150
    IC=IC+1
    SV(J)=SV(IC)
    SV(IC)=I
    UNDER(I)=1
    GO TO 200
150 CONTINUE
    ITAL=ITAL+1
    IC=IC+1
    SV(ITAL)=SV(IC)
    SV(IC)=I
    YS(I)=1
    YBEST(I)=1
    SS(I)=1
200 CONTINUE
    WRITE(6,21) IC
    LTRN=0
    DO 300 J=1,L
    ZFIX(J)=CONST(J)
    DO 250 I=1,NOSRCS
250 IF(SS(I).GT.0.AND.YBEST(I).GT.0) ZFIX(J)=ZFIX(J)+CV(J,I)
300 CONTINUE
    GO TO 7000
1000 SUMFRE=0
    LBND=FXS+MINTRN+LTRN
    IF(LBND.GE.ZPLUS) GO TO 7000
    FRE=0
    ZMAX=-LARGE
    DO 1100 J=1,L
    IZFREE=ZFIX(J)
    DO 1050 I=1,NOSRCS
    IF(SS(I).GT.0) GO TO 1050
    YS(I)=0
    II=LBND+FIX(I)
    IF(CV(J,I).GE.0.OR.II.GE.ZPLUS) GO TO 1050
    IZFREE=IZFREE+CV(J,I)
    IF(IZFREE.LT.ZMAX) GO TO 1100
    YS(I)=1
1050 CONTINUE
    ZMAX=IZFREE
    IF(ZMAX.GE.ZPLUS) GO TO 7000
    JMAX=J
    DO 1060 I=1,NOSRCS
1060 YBEST(I)=YS(I)
1100 CONTINUE
    DO 1180 I=1,NOSRCS
```

```
      IF(SS(I).NE.0.OR.YBEST(I).LE.0) GO TO 1180
      SUMFRE=SUMFRE+AVAIL(I)
      FRE=FRE+FIX(I)
1180  CONTINUE
2000  II=SUMMI+SUMFRE
      IF(II.GE.SUMDJ) GO TO 4000
3000  AMAX=LARGE
      SUMTS=0
      DO 3020 I=1,NOSRCS
      III=FXS+FRE+MINTRN+FIX(I)
      IF(SS(I).GT.0.OR.YBEST(I).GT.0.OR.III.GE.ZPLUS) GO TO 3020
      SUMTS=SUMTS+AVAIL(I)
      A=FIX(I)/AVAIL(I)
      IF(A.GE.AMAX) GO TO 3020
      AMAX=A
      INGT=I
3020  CONTINUE
      II=SUMMI+SUMFRE+SUMTS
      IF(II.LT.SUMDJ) GO TO 7000
      ITAL=ITAL+1
      SV(ITAL)=INGT
      NOLP=.FALSE.
      SS(INGT)=1
      FXS=FXS+FIX(INGT)
      YBEST(INGT)=1
      YS(INGT)=1
      SUMMI=SUMMI+AVAIL(INGT)
      DO 3050 J=1,L
3050  ZFIX(J)=ZFIX(J)+CV(J,INGT)
      II=SUMMI+SUMFRE
      IF(II.LT.SUMDJ) GO TO 3000
      GO TO 1000
4000  ZBMAX=-LARGE
      DO 4010 J=1,L
      ZBAR(J)=ZFIX(J)
      DO 4005 I=1,NOSRCS
      IF(SS(I).EQ.0.AND.YBEST(I).GT.0) ZBAR(J)=ZBAR(J)+CV(J,I)
4005  CONTINUE
      IF(ZBAR(J).LE.ZBMAX) GO TO 4010
      ZBMAX=ZBAR(J)
      JBAR=J
4010  CONTINUE
      IF(ZMAX.EQ.ZBMAX) GO TO 6000
5000  ITMIN=LARGE
      DO 5040 I=1,NOSRCS
      IF(SS(I).GT.0) GO TO 5040
      IF(YBEST(I).NE.1.OR.CV(JBAR,I).LE.0) GO TO 5020
      ITMAX=-LARGE
      DO 5010 J=1,L
      II=ZBAR(J)-CV(J,I)
      IF(II.GT.ITMAX) ITMAX=II
5010  CONTINUE
      IF(ITMAX.GE.ITMIN) GO TO 5040
      ITMIN=ITMAX
```

```
      IMIN=I
      SETONE=.FALSE.
      GO TO 5040
 5020 II=FXS+FRE+MINTRN+FIX(I)
      IF(YBEST(I).NE.0.OR.CV(JBAR,I).GE.0.OR.II.GE.ZPLUS) GO TO 5040

      ITMAX=-LARGE
      DO 5030 J=1,L
      II=ZBAR(J)+CV(J,I)
      IF(II.GT.ITMAX) ITMAX=II
 5030 CONTINUE
      IF(ITMAX.GE.ITMIN) GO TO 5040
      ITMIN=ITMAX
      IMIN=I
      SETONE=.TRUE.
 5040 CONTINUE
      IF(ITMIN.NE.LARGE) GO TO 5210
      IF(ZBMAX.GE.ZPLUS) GO TO 7000
C IF NO SURROGATE CONSTRAINTS ARE DESIRED A GO TO 6000 IS NEEDED
      III=0
      IF(NOLP) GO TO 6000
      IF(ITAL.GE.III) GO TO 6000
      CALL LINEAR
      NOLP=.TRUE.
      IF(BACK) GO TO 7000
      ICTR=ICTR+1
      IF(ICTR.LT.100) GO TO 5050
      LL=MOD(ICTR,100)+1
      GO TO 5060
 5050 LL=L+1
 5060 SUM=0.
      DO 5070 J=1,L
 5070 IF(U(J).GT.0.) SUM=SUM+U(J)*CONST(J)
      CONST(LL)=SUM
      DO 5090 I=1,NOSRCS
      SUM=0.
      DO 5080 J=1,L
 5080 IF(U(J).GT.0.) SUM=SUM+U(J)*CV(J,I)
 5090 CV(LL,I)=SUM
      ZFIX(LL)=CONST(LL)
      DO 5100 I=1,NOSRCS
 5100 IF(SS(I).GT.0.AND.YBEST(I).GT.0) ZFIX(LL)=ZFIX(LL)+CV(LL,I)
      GO TO 1000
 5210 CONTINUE
      ITAL=ITAL+1
      NOLP=.FALSE.
      SS(IMIN)=1
      IF(SETONE) GO TO 5220
      SV(ITAL)=-IMIN
      FRE=FRE-FIX(IMIN)
      SUMFRE=SUMFRE-AVAIL(IMIN)
      LTRN=LTRN+LOCMIN(IMIN)
      YS(IMIN)=0
      YBEST(IMIN)=0
      GO TO 1000
```

```
5220 SV(ITAL)=IMIN
     FXS=FXS+FIX(IMIN)
     SUMMI=SUMMI+AVAIL(IMIN)
     YS(IMIN)=1
     YBEST(IMIN)=1
     DO 5250 J=1,L
5250 ZFIX(J)=ZFIX(J)+CV(J,IMIN)
     GO TO 1000
6000 FXCOST=FXS+FRE
     DO 6010 I=1,NOSRCS
     IF(YBEST(I).EQ.0) GO TO 6005
     HI(I)=AVAIL(I)
     GO TO 6010
6005 HI(I)=0
6010 CONTINUE
     CALL NETWRK
     IF(NTFEAS) GO TO 8999
     ITT=0
     DO 6020 I=1,NOSRCS
     IF(F(I).NE.0.OR.YBEST(I).EQ.0) GO TO 6020
     ITT=ITT+FIX(I)
6020 CONTINUE
     ZSTAR=FXCOST+ISUM
     II=ZSTAR-ITT
     IF(II.GE.ZPLUS) GO TO 6500
     FCPLUS=FXCOST-ITT
     ZPLUS=II
     TCPLUS=ISUM
     DO 6030 I=1,NOSRCS
     IF(F(I).EQ.0) GO TO 6025
     YPLUS(I)=1
     GO TO 6030
6025 YPLUS(I)=0
6030 CONTINUE
     IF(ZPLUS.LE.ZBMAX) GO TO 7000
6500 ICTR=ICTR+1
     NOTIME=0
     IF(ICTR.GT.100) GO TO 6505
     L=ICTR
     LONE=L
     GO TO 6510
6505 LONE=MOD(ICTR,100)+1
6510 CONST(LONE)=ISMJ
     ZFIX(LONE)=ISMJ
     DO 6520 I=1,NOSRCS
     CV(LONE,I)=FIX(I)-(PI(I)*AVAIL(I))
     IF(SS(I).GT.0.AND.YBEST(I).GT.0)
    *    ZFIX(LONE)=ZFIX(LONE)+CV(LONE,I)
6520 CONTINUE
     IF(FIRST) GO TO 25
     GO TO 1000
7000 I=ITAL
     NOLP=.FALSE.
7010 II=IABS(SV(I))
```

```
      IF(UNDER(II).NE.0) GO TO 7070
      IF(I.EQ.ITAL) GO TO 7040
      III=I+1
      DO 7035 K=III,ITAL
      IF(SV(K).LT.0) GO TO 7030
      II=SV(K)
      SS(II)=0
      UNDER(II)=0
      SV(K)=0
      YS(II)=0
      YBEST(II)=0
      FXS=FXS-FIX(II)
      DO 7020 J=1,L
7020  ZFIX(J)=ZFIX(J)-CV(J,II)
      SUMMI=SUMMI-AVAIL(II)
      GO TO 7035
7030  II=-SV(K)
      SS(II)=0
      UNDER(II)=0
      SV(K)=0
      LTRN=LTRN-LOCMIN(II)
7035  CONTINUE
7040  CONTINUE
      IF(SV(I).LT.0) GO TO 7050
      II=SV(I)
      YS(II)=0
      YBEST(II)=0
      FXS=FXS-FIX(II)
      DO 7045 J=1,L
7045  ZFIX(J)=ZFIX(J)-CV(J,II)
      SUMMI=SUMMI-AVAIL(II)
      LTRN=LTRN+LOCMIN(II)
      GO TO 7065
7050  II=-SV(I)
      YS(II)=1
      YBEST(II)=1
      FXS=FXS+FIX(II)
      DO 7060 J=1,L
7060  ZFIX(J)=ZFIX(J)+CV(J,II)
      SUMMI=SUMMI+AVAIL(II)
      LTRN=LTRN-LOCMIN(II)
7065  CONTINUE
      UNDER(II)=1
      SV(I)=-SV(I)
      ITAL=I
7068  CONTINUE
      II=MINTRN+FXS+LTRN
      IF(II.GE.ZPLUS) GO TO 7000
      GO TO 1000
7070  CONTINUE
      I=I-1
      IF(I.LE.IC) GO TO 7080
      GO TO 7010
7080  CONTINUE
```

```
      CALL WRTOUT
      GO TO 9999
8999  WRITE(6,9001)
9001  FORMAT(2X,'THE PROBLEM IS NOT FEASABLE')
9999  CONTINUE
      STOP
      END
```

- - - T H E E N D - - -

```
      SUBROUTINE NETWRK
      COMMON ITER,ISMJ,ICTR,BACK,SUMMI,SUMDJ,YBEST(30)
      COMMON  NOSRCS,NODEST,NODES,ARCS,ISUM,LARGE,ZPLUS,TCPLUS,FCPLUS,
     *        YPLUS(30),COST(1376),F(1376),HI(1376),PI(79),FIX(30),
     *        AVAIL(30),DEM(50)
      COMMON NTFEAS
      COMMON L,ITAL,SS(30),CONST(100),CV(100,30),U(100),ITX
      INTEGER SS,CONST,CV
      LOGICAL NTFEAS
      INTEGER NOSRCS,NODEST,NODES,ARCS,ISUM,LARGE,ZPLUS,TCPLUS,FCPLUS,
     *        YPLUS,F,HI,PI,FIX,AVAIL,DEM,COST
      INTEGER S(1376),T(1376),LO(1376),G(1376),H(1376),UU(79),V(79),
     *        R(79),LL(79),A,AA,EPS,EPS1,TERM,ORIGIN,SSS,P,REFNOD
      LOGICAL BRKTHR
      ITER=ITER+1
      IF(ITER.GT.1) GO TO 50
      REFNOD=NODES
      DO 10 J=1,NOSRCS
      S(J)=NODES
      T(J)=J
      HI(J)=AVAIL(J)
   10 CONTINUE
      K=NOSRCS
      DO 20 I=1,NOSRCS
      DO 15 J=1,NODEST
      K=K+1
      S(K)=I
      T(K)=J+NOSRCS
      HI(K)=LARGE
   15 CONTINUE
   20 CONTINUE
      II=NOSRCS+NOSRCS*NODEST
      IN=II+NODEST
      ITL=NOSRCS
      ISD=NODES-1
      ITRK=0
      II=II+1
      DO 30 K=II,IN
      ITL=ITL+1
      ITRK=ITRK+1
      S(K)=ITL
      T(K)=ISD
      HI(K)=DEM(ITRK)
   30 CONTINUE
      S(ARCS)=ISD
      T(ARCS)=NODES
      HI(ARCS)=LARGE
      COST(ARCS)=-LARGE
      DO 45 I=II,IN
   45 COST(I)=0
      NN=NODES+2
      DO 3 I=3,NN
      UU(I)=0
      V(I)=0
```

```
  3 CONTINUE
    DO 4 J=1,ARCS
    I=S(J)+2
    II=T(J)+2
    UU(I)=UU(I)+1
    V(II)=V(II)+1
    LO(J)=0
    F(J)=0
  4 CONTINUE
    DO 5 I=1,NODES
    PI(I)=0
  5 CONTINUE
    DO 6 I=1,2
    UU(I)=1
    V(I)=1
  6 CONTINUE
    NN=NODES+1
    DO 7 I=3,NN
    UU(I)=UU(I-1)+UU(I)
    V(I)=V(I-1)+V(I)
  7 CONTINUE
    DO 8 J=1,ARCS
    II=S(J)+1
    III=UU(II)
    G(III)=J
    UU(II)=UU(II)+1
    II=T(J)+1
    III=V(II)
    H(III)=J
    V(II)=V(II)+1
  8 CONTINUE
    NN=NODES+2
    LSTARC=    ARCS
 50 DO 59 J=1,ARCS
    II=S(J)
    JJ=T(J)
    COST(J)=COST(J)+PI(II)-PI(JJ)
 59 CONTINUE
    IF(ITER.GT.2) LSTARC=NOSRCS
    JJ=1
    EPS=999999999
    AA=0
    BRKTHR=.TRUE.
115 IF(F(JJ).LT.LO(JJ).OR.COST(JJ).LT.0.AND.F(JJ).LT.HI(JJ))
   *    GO TO 125
    IF(F(JJ).GT.HI(JJ).OR.COST(JJ).GT.0.AND.F(JJ).GT.LO(JJ))
   *    GO TO 130
    SMALL=.00000005
    SON=ABS(COST(JJ))
    IF(LO(JJ).EQ.HI(JJ).AND.SON.GT.SMALL.AND.BRKTHR.
   *    AND.EPS.NE.0) GO TO 125
    JJ=JJ+1
    EPS=999999999
    IF(JJ.LE.LSTARC) GO TO 115
    DO 122 J=1,ARCS
    II=S(J)
```

```
      III=T(J)
      COST(J)=COST(J)+PI(III)-PI(II)
  122 CONTINUE
      GO TO 300
  125 TERM=S(JJ)
      ORIGIN=T(JJ)
      LABORG=JJ
      GO TO 135
  130 TERM=T(JJ)
      ORIGIN=S(JJ)
      LABORG=-JJ
  135 R(1)=ORIGIN
  140 CONTINUE
      IF(JJ.NE.AA) GO TO 141
      IF(BRKTHR) GO TO 141
      GO TO 145
  141 CONTINUE
      DO 142 I=1,NODES
      LL(I)=0
  142 CONTINUE
      SSS=1
  145 P=1
      AA=JJ
      BRKTHR=.FALSE.
      LL(ORIGIN)=LABORG
  150 I=R(P)
      II=UU(I)
      IN=UU(I+1)-1
      DO 151 A=II,IN
      J=G(A)
      K=T(J)
      IF(LL(K).NE.0) GO TO 151
      IF(F(J).LT.LO(J).OR.COST(J).LE.0.AND.F(J).LT.HI(J)) GO TO 152

      GO TO 151
  152 LL(K)=J
      SSS=SSS+1
      R(SSS)=K
  151 CONTINUE
      II=V(I)
      IN=V(I+1)-1
      DO 153 A=II,IN
      J=H(A)
      K=S(J)
      IF(LL(K).NE.0) GO TO 153
      IF(F(J).GT.HI(J).OR.COST(J).GE.0.AND.F(J).GT.LO(J)) GO TO 154

      GO TO 153
  154 LL(K)=-J
      SSS=SSS+1
      R(SSS)=K
  153 CONTINUE
      IF(LL(TERM).NE.0) GO TO 155
      P=P+1
      IF(P.GT.SSS) GO TO 190
      GO TO 150
```

```
155 EPS=999999999
    BRKTHR=.TRUE.
    KT=TERM
    J=1
160 KQ=LL(KT)
    KP=IABS(KQ)
    IF(KQ.GT.0) GO TO 165
    KT=T(KP)
    IF(COST(KP).GE.0) GO TO 175
    GO TO 170
165 KT=S(KP)
    IF(COST(KP).GT.0) GO TO 175
170 IB=IABS(HI(KP)-F(KP))
    IF(EPS.GT.IB) EPS=IB
    GO TO 180
175 IB=IABS(LO(KP)-F(KP))
    IF(EPS.GT.IB) EPS=IB
180 R(J)=KQ
    IF(KT.EQ.TERM) GO TO 185
    J=J+1
    GO TO 160
185 DO 188 I=1,J
    IF(R(I).LE.0) GO TO 187
    II=R(I)
    F(II)=F(II)+EPS
    GO TO 188
187 II=-R(I)
    F(II)=F(II)-EPS
188 CONTINUE
    GO TO 115
190 EPS1=999999999
    DO 192 J=1,ARCS
    II=S(J)
    III=T(J)
    IF(LL(II).NE.0.AND.LL(III).EQ.0.AND.F(J).LT.HI(J).OR.
   *   LL(II).EQ.0.AND.LL(III).NE.0.AND.F(J).GT.LO(J)) GO TO 191
    GO TO 192
191 IB=IABS(COST(J))
    IF(EPS1.GT.IB) EPS1=IB
192 CONTINUE
    EPS=EPS1
    IF(EPS.NE.999999999) GO TO 195
    IF(ABS(COST(JJ)).LT.SMALL) GO TO 400
    IF(LL(ORIGIN).GE.0.AND.COST(JJ).GT.0) GO TO 400
    IF(LL(ORIGIN).LT.0.AND.COST(JJ).LT.0) GO TO 400
    EPS=IABS(COST(JJ))
195 DO 995 J=1,ARCS
    II=S(J)
    III=T(J)
    IF(LL(II).EQ.0.AND.LL(III).NE.0) COST(J)=COST(J)+EPS
    IF(LL(II).NE.0.AND.LL(III).EQ.0) COST(J)=COST(J)-EPS
995 CONTINUE
    IF(LL(REFNOD).EQ.0) GO TO 197
    DO 196 I=1,NODES
```

```
      IF(LL(I).EQ.0) PI(I)=PI(I)+EPS
  196 CONTINUE
      GO TO 199
  197 DO 198 I=1,NODES
      IF(LL(I).NE.0) PI(I)=PI(I)-EPS
  198 CONTINUE
  199 IF(EPS.EQ.EPS1.OR.F(JJ).EQ.LO(JJ).OR.F(JJ).EQ.HI(JJ)) GO TO 115

  300 ISUM=0
      II=NOSRCS+1
      III=ARCS-NODEST-1
      DO 310 I=II,III
      ISUM=ISUM+F(I)*COST(I)
  310 CONTINUE
      IF(ITER.EQ.1) GO TO 420
      ISMI=0
      DO 315 I=1,NOSRCS
      ISMI=ISMI+PI(I)*HI(I)
  315 CONTINUE
      ISMJ=0
      II=NODES-2
      III=NOSRCS+1
      K=NOSRCS+NOSRCS*NODEST
      DO 320 J=III,II
      K=K+1
      ISMJ=ISMJ+PI(J)*HI(K)
  320 CONTINUE
      II=ISMJ-ISMI
      IF(II.NE.ISUM) GO TO 410
      GO TO 420
  410 WRITE(6,411)
  411 FORMAT(' INCORRECT DUALS')
  400 CONTINUE
      NTFEAS=.TRUE.
  420 CONTINUE
      RETURN
      END
```

- - - T H E E N D - - -

```
      SUBROUTINE LINEAR
      COMMON ITER,ISMJ,ICTR,BACK,SUMMI,SUMDJ,YBEST(30)
      COMMON  NOSRCS,NODEST,NODES,ARCS,ISUM,LARGE,ZPLUS,TCPLUS,FCPLUS,
     *        YPLUS(30),COST(1376),F(1376),HI(1376),PI(79),FIX(30),
     *        AVAIL(30),DEM(50)
      COMMON NTFEAS
      COMMON L,ITAL,SS(30),CONST(100),CV(100,30),U(100),ITX
      DIMENSION D(30,160),P(160),IBV(160),SC(160),IYF(30)
      INTEGER SS,CONST,CV,COST
      INTEGER ZPLUS,TCPLUS,FCPLUS,AVAIL,SUMMI,SUMDJ,YBEST,YPLUS,FIX

      LOGICAL BACK
      BACK=.FALSE.
      ITX=ITX+1
      IWW=NOSRCS-ITAL
      IW=IWW+1
      IY=L+IW
      IZ=L+2*IW
      IX=IZ-1
      DO 140 J=1,L
      K=0
      P(J)=CONST(J)
      D(IW,J)=1.
      DO 140 I=1,NOSRCS
      IF(SS(I).GT.0) GO TO 130
      K=K+1
      D(K,J)=-CV(J,I)
      GO TO 140
  130 P(J)=P(J)+CV(J,I)*YBEST(I)
  140 CONTINUE
      ABC=-1.
      ABCD=-1.
      II=L+1
      III=L+IWW
  150 K=0
      DO 170 J=II,III
      K=K+1
      P(J)=ABC
      DO 170 I=1,IW
      IF(I.EQ.K) GO TO 160
      D(I,J)=0.
      GO TO 170
  160 D(I,J)=ABCD
      IF(III.EQ.IX) IBV(I)=J
  170 CONTINUE
      IF(III.EQ.IX) GO TO 180
      II=III+1
      III=IX
      ABC=0.
      ABCD=1.
      GO TO 150
  180 CONTINUE
      DO 190 I=1,IWW
  190 D(I,IZ)=0.
      P(IX)=-999999.
```

```
      D(IW,IZ)=1.
  220 CONTINUE
      SMALL=0.0000005
      Z=0.
      DO 230 M=1,IW
      IBVM=IBV(M)
  230 Z=Z+D(M,IZ)*P(IBVM)
      NOPIVS=0
  240 SCMAX=0.
      DO 270 N=1,IX
      DO 250 I=1,IW
      IF(N.EQ.IBV(I)) GO TO 270
  250 CONTINUE
      SUM=0.
      DO 260 I=1,IW
      J=IBV(I)
  260 SUM=SUM+P(J)*D(I,N)
      SC(N)=P(N)-SUM
      IF(SC(N).LE.SCMAX) GO TO 270
      SCMAX=SC(N)
      IPIVCO=N
  270 CONTINUE
      DO 280 M=1,IW
      IBVM=IBV(M)
  280 SC(IBVM)=0.
      IF(SCMAX.LE.0) GO TO 380
      NOPIVS=NOPIVS+1
  290 SMLVAL=999999999.
      DO 320 M=1,IW
      IF(D(M,IPIVCO)) 320,320,300
  300 QUONT=D(M,IZ)/D(M,IPIVCO)
      IF(QUONT-SMLVAL) 310,320,320
  310 IPIVRO=M
      SMLVAL=QUONT
  320 CONTINUE
      IBV(IPIVRO)=IPIVCO
      DIV=D(IPIVRO,IPIVCO)
      DO 330 N=1,IZ
      CRANK=D(IPIVRO,N)
  330 D(IPIVRO,N)=CRANK/DIV
  340 DO 370 M=1,IW
      IF(M-IPIVRO) 350,370,350
  350 CM=-D(M,IPIVCO)
      DO 360 N=1,IZ
      TM=D(IPIVRO,N)*CM
      SINKS=D(M,N)
  360 D(M,N)=SINKS+TM
  370 CONTINUE
      Z=Z+SMLVAL*SCMAX
      IF(Z.LT.ZPLUS) GO TO 240
      WRITE(6,375)
  375 FORMAT(2X,'BACKTRACK...NO BETTER SOLUTION')
      BACK=.TRUE.
      RETURN
```

```
380 CONTINUE
    II=L+2
    III=II+IW
    K=0
    DO 390 I=1,IW
390 IYF(I)=0
    DO 410 I=II,III
    ABSC=ABS(SC(I))
    IF(ABSC.GE.SMALL) GO TO 400
    K=K+1
    IYF(K)=0
    GO TO 410
400 ASC=SC(I)-1.
    ABSC=ABS(ASC)
    IF(ABSC.GE.SMALL) GO TO 450
    K=K+1
    IYF(K)=1
410 CONTINUE
    K=0
    BACK=.TRUE.
    ZPLUS=Z
    DO 430 I=1,NOSRCS
    IF(SS(I).GT.0) GO TO 420
    K=K+1
    YPLUS(I)=IYF(K)
    GO TO 430
420 YPLUS(I)=YBEST(I)
430 CONTINUE
    FCPLUS=0
    DO 440 I=1,NOSRCS
440 IF(YPLUS(I).GT.0) FCPLUS=FCPLUS+FIX(I)
    TCPLUS=ZPLUS-FCPLUS
    WRITE(6,445)
445 FORMAT(2X,'BACKTRACK...BETTER SOLUTION FOUND')
    RETURN
450 CONTINUE
    DO 460 J=1,L
460 U(J)=0
    DO 470 I=1,IW
    J=IBV(I)
    IF(J.LE.L) U(J)=D(I,IZ)
470 CONTINUE
    DO 405 J=1,L
405 CONTINUE
    WRITE(6,407)
407 FORMAT(2X,'A SURROGATE CONSTRAINT HAS BEEN FOUND')
    RETURN
    END
```

- - - T H E E N D - - -

```
      SUBROUTINE WRTOUT
      COMMON ITER,ISMJ,ICTR,BACK,SUMMI,SUMDJ,YBEST(30)
      COMMON  NOSRCS,NODEST,NODES,ARCS,ISUM,LARGE,ZPLUS,TCPLUS,FCPLUS,
     *        YPLUS(30),COST(1376),F(1376),HI(1376),PI(79),FIX(30),
     *        AVAIL(30),DEM(50)
      COMMON NTFEAS
       COMMON L,ITAL,SS(30),CONST(100),CV(100,30),U(100),ITX
       INTEGER SS,CONST,CV
      LOGICAL NTFEAS
      INTEGER NOSRCS,NODEST,NODES,ARCS,ISUM,LARGE,ZPLUS,TCPLUS,FCPLUS,
     *        YPLUS,F,HI,PI,FIX,AVAIL,DEM
      WRITE(6,8) ICTR
      WRITE(6,9) ITX
      WRITE(6,7) ITER
      WRITE(6,1)
      DO 2 I=1,NOSRCS
      IF(YPLUS(I).EQ.0) GO TO 2
      WRITE(6,3) I
    2 CONTINUE
      WRITE(6,4) TCPLUS
      WRITE(6,5) FCPLUS
      WRITE(6,6) ZPLUS
    1 FORMAT(2X,'THE FOLLOWING ARE OPENED PLANTS ')
    3 FORMAT(2X,'PLANT ',I4)
    4 FORMAT(2X,'THE TRANSPORTATION COST IS ',I12)
    5 FORMAT(2X,'THE FIXED COST IS ',I12)
    6 FORMAT(2X,'THE TOTAL COST IS ',I15)
    7 FORMAT(2X,'THE NUMBER OF NETWORKS SOLVED IS ',I6)
    8 FORMAT(2X,'THE NUMBER OF CONSTRAINTS ADDED IS',I6)
    9 FORMAT(2X,'THE NUMBER OF LP"S SOLVED IS ',I6)
      RETURN
      END
```

- - - T H E E N D - - -

APPENDIX C

TEST PROBLEM DATA

Transportation Costs

The problems which were used to test the algorithm have all appeared in the literature. Problem one and problems seven through sixteen are based on the Kuehn and Hamburger problems, but with an additional source at the factory (Indianapolis). The first problem uses the first ten sources and twenty destinations mentioned. Problems seven through twelve use the first sixteen sources (including Indianapolis) and all destinations. Problems thirteen through sixteen use all sources (including Indianapolis) and all destinations.

The transportation cost, c_{ij}, is then given by rounding the following expression to the nearest cent.

$$c_{ij} = (1.25)\bar{d}_i + (2.5)d_{ij}$$

The d_{ij} refer to the railroad distance between plant i and destination j, and $\bar{d}_i$ is the distance from Indianapolis to plant i.†

Problems two through six appear in Ellwein, and have fifteen plants and forty-five destination. The transportation cost (in dollars) is given in Table 3. The columns correspond to plants, and each row represents a destination.

---

†Railroad distances were obtained from the Rand McNally Commercial Atlas and Marketing Guide, p. 17, Rand McNally and Company, Chicago, 1969.

Table 3. Transportation Cost for Problems 2-6

10	25	50	50	50	999	999	999	999	999	999	999	50	999	50
10	25	50	50	50	999	999	999	999	999	999	999	999	999	50
10	25	50	50	50	999	999	999	999	999	999	999	999	999	50
10	25	50	50	50	999	999	999	999	999	999	999	999	999	50
3	10	50	50	25	999	999	999	999	999	999	999	999	50	50
10	25	50	50	25	50	999	999	999	999	999	999	999	999	999
3	10	25	30	25	50	999	999	999	999	999	999	50	50	50
3	10	25	30	25	50	50	999	999	999	999	999	999	50	999
3	10	25	10	25	50	50	999	999	50	50	50	50	50	25
1	10	25	10	25	50	50	999	999	50	50	50	999	50	999
3	10	25	10	25	50	50	50	999	999	999	999	999	50	999
10	10	25	10	25	50	50	50	999	999	999	999	999	999	999
999	10	25	10	25	50	50	50	999	999	50	50	50	999	999
1	3	25	10	25	25	25	50	50	50	50	50	50	50	999
0	1	10	5	10	25	25	25	50	50	50	50	50	50	25
5	1	10	5	25	25	25	25	50	50	50	50	999	999	25
5	0	3	5	10	10	25	25	50	25	25	50	50	30	25
5	1	1	5	5	10	25	25	50	25	25	50	50	25	15
10	3	0	5	5	10	10	10	50	25	25	50	25	25	15
10	3	1	0	5	5	10	25	25	15	25	50	50	25	50
10	3	1	5	0	5	10	10	25	15	20	25	25	25	10
25	25	3	5	5	5	10	5	25	15	15	25	25	20	10
25	25	3	5	5	0	5	10	25	15	15	25	25	25	25
25	25	10	5	10	5	0	10	25	15	15	25	25	50	50
999	25	10	20	10	5	10	5	15	10	15	20	20	20	15
999	25	50	20	10	10	10	0	5	10	15	20	20	15	10
999	25	50	20	10	5	10	5	5	10	15	20	15	25	30
999	999	50	50	10	25	25	5	10	10	10	10	15	10	10
999	999	999	50	25	10	25	5	5	10	10	10	10	15	15
999	999	999	50	50	10	25	10	5	10	5	5	5	10	15
999	999	999	50	25	25	25	5	5	5	10	10	20	20	20
999	999	999	50	50	25	25	10	0	5	5	5	10	20	15
999	999	999	50	25	25	25	10	5	0	5	10	15	20	25
999	999	999	50	999	50	50	25	5	5	0	5	5	20	15
999	999	999	50	999	50	50	25	10	10	5	0	5	20	15
999	999	999	999	999	50	50	25	10	15	10	5	0	10	15
999	999	999	999	999	50	50	25	10	10	5	5	10	20	25
999	999	999	999	999	999	999	50	50	15	15	15	20	20	50
999	999	999	999	999	999	999	50	25	20	15	15	15	10	25
999	999	999	999	999	999	999	999	999	50	25	25	20	5	20
999	999	999	999	999	999	999	999	50	50	25	25	20	5	10
999	999	999	999	999	999	999	25	50	50	20	25	10	0	10
999	999	20	50	25	25	50	25	10	25	15	10	5	5	5
999	999	20	999	50	50	50	25	25	25	15	20	5	5	5
999	999	20	999	25	25	50	25	25	25	20	20	15	10	0

Fixed Costs (in thousands)

Problem 1:

All fixed costs equal 12.5

Problem 2:

30	17.5	22.5	20	35	27.5	40	
15	17.5	30	22.5	40	20	17.5	

Problem 3:

45	25.5	34	30	52.5	41.5	60	
22.5	25.5	45	34	34	60	30	25.5

Problem 4:

Same as Problem 3.

Problem 5:

Same as Problem 3

Problem 6:

The fixed cost of Problem 3 multiplied by 10.

Problem 7:

All fixed costs equal 12.5

Problem 8:

All fixed costs equal 17.5 except the 11^{th}, which is zero.

Problem 9

All fixed costs equal 12.5 except the 11^{th}, which is zero.

Problem 10:

Same as Problem 9.

Problem 11:

Same as Problem 8.

Problem 12:

12	25	25	40	12	25	12	12
40	12	8	12	40	12	12	12

Problem 13:

All fixed costs equal 7.5 except the 11^{th}, which is zero.

Problem 14:

92	36	62	86	93	86	11	35	
60	28	35	56	95	41	66	88	
99	43	15	86	79	33	38	29	58

Problem 15:

Same as Problem 14.

Problem 16:

The fixed costs of Problem 14 multiplied by 10.

Plant Capacities (in thousands)

Problem 1:

All capacities equal 5.

Problem 2:

5	2.5	2.5	1	5	2.5	5	1
2.5	5	2.5	2.5	5	1	2.5	

Problem 3:

7.5	3.75	3.75	1.5	7.5	3.75	7.5	1.5
3.75	7.5	3.75	3.75	7.5	1.5	3.75	

Problem 4:

10	10	5	10	7.5	5	10	4
3.75	7.5	5	5	7.5	10	5	

Problem 5:

No capacity constraints.

Problem 6:

Same as Problem 4.

Problem 7:

All capacities equal 5.

Problem 8:

Same as Problem 7.

Problem 9:

All capacities equal 15.

Problem 10:

No capacity constraints

Problem 11:

No capacity constraints

Problem 12:

8	12	12	24	8	12	8	8	24
8	18	8	24	8	8	8		

Problem 13:

No capacity constraints.

Problem 14:

8	10	10	18	4	10	4	4	15
4	16	4	14	4	4	4	18	14
10	10	10	4	12	4	10		

Problem 15:

16	24	24	40	12	24	12	12	32
12	32	12	32	12	12	12	40	32
24	24	24	12	24	12	24		

Problem 16:

Same as Problem 15.

Destination Demands

Problems 2-6:

733	1746	826	908	1135	1402	1264
741	823	1103	435	806	418	921
1945	571	841	1001	1421	361	1725
316	582	281	944	961	371	551
1479	666	730	1407	1090	1617	1087
1196	1584	1118	227	30	35	27
3	4	8				

Problems 1, 7-16:

146	87	672	1337	31	559	2370
1089	33	32	5495	904	1466	143
615	564	226	3016	253	195	38
807	551	304	814	337	4368	577
482	495	231	322	685	12912	325
366	3671	2213	705	328	1681	1117
275	500	2241	733	222	49	1464
222						

It should be noted that the configuration constraints which were added by Ellwein on Problems 4-6, 12 and 14-16 were ignored.

BIBLIOGRAPHY

1. Benders, J. F., "Partitioning Procedures for Solving Mixed-Variables Programming Problems," Numerische Mathematik, Vol. 4, pp. 238-252 (1962).

2. Bradley, G. H., "Transformation of Integer Programs for Knapsack Problems," Yale University Department of Administration Sciences Report No. 37 (1970).

3. Davis, P. S. and T. L. Ray, "A Branch-Bound Algorithm for the Capacitated Facilities Location Problem," Naval Research Logistics Quarterly, Vol. 16, pp. 331-344 (1969).

4. Efroymson, M. A. and T. L. Ray, "A Branch Bound Algorithm for Plant Location," Operations Research, Vol. 14, pp. 361-368 (1966).

5. Ellwein, L. B., "Fixed Charge Location - Allocation Problems with Capacity and Configuration Constraints," Stanford University Department of Industrial Engineering Technical Report 70-2 (1970).

6. Feldman, E., F. A. Lehrer and T. L. Ray, "Warehouse Location under Continuous Economies of Scale," Management Science, Vol. 12, pp. 670-684 (1966).

7. Ford, L. R. and D. R. Fulkerson, Flows in Networks, Princeton University Press, Princeton, N. J. (1962).

8. Geoffrion, A. M., "Integer Programming by Implicit Enumeration and Balas' Method," SIAM Review, Vol. 9, pp. 178-190 (1967).

9. Geoffrion, A. M., "An Improved Implicit Enumeration Approach for Integer Programming," Rand Memorandum RM-5644-PR, June (1968).

10. Glover, F. "Surrogate Constraints," Operations Research, Vol. 16, pp. 741-749 (1968).

11. Gray, P., "Mixed Integer Programming Algorithms for Site Selection and Other Fixed Charge Problems having Capacity Constraints," Stanford Research Institute (SED) Special Report (1967).

12. Kuehn, A. A. and M. J. Hamburger, "A Heuristic Program for Locating Warehouses," Management Science, Vol. 9, pp. 643-666 (1963).

13. Lawler, E. L. and D. E. Wood, "Branch-and-Bound Methods: A Survey," Operations Research, Vol. 14, pp. 699-719 (1966).

14. Manne, A. S., "Plant Location Under Economies of Scale-Decentralization and Computation," Management Science, Vol. 11, pp. 213-235 (1964).

15. Marks, D. H., "Facility Location and Routing Models in Solid Waste Collection Systems," Ph. D. Thesis, The Johns Hopkins University (1969).

16. Sa, G., "Branch-and-Bound and Approximate Solutions to the Capacitated Plant-Location Problem," Operations Research, Vol. 17, pp. 1005-1016 (1969).

17. Spielberg, K., "Algorithms for the Simple Plant Location Problem with Some Side Conditions," Operations Research, Vol. 17, pp. 85-111 (1969).

18. Tomlin, J. A., "Branch and Bound Methods for Integer and Non-Convex Programming," in J. Abadie (ed.), Integer and Nonlinear Programming, North-Holland, Amsterdam (1970).

19. Unger, V. E., "Capital Budgeting and Mixed Zero-One Integer Programming," AIIE Transactions, Vol. II, pp. 28-36 (1970).